Mobile Security with Location-Aware Role-Based Access Control

Nils Ulltveit-Moe and Vladimir Oleshchuk

University of Agder, Service Box 509, 4898 Grimstad, Norway
{nils.ulltveit-moe,vladimir.oleshchuk}@uia.no

Abstract. This paper describes how location-aware Role-Based Access Control (RBAC) can be implemented on top of the Geographically eXtensible Access Control Markup Language (GeoXACML). It furthermore sketches how spatial separation of duty constraints (both static and dynamic) can be implemented using GeoXACML on top of the XACML RBAC profile. The solution uses physical addressing of geographical locations which facilitates easy deployment of authorisation profiles to the mobile device. Location-aware RBAC can be used to implement location dependent access control and also other security enhancing solutions on mobile devices, like location dependent device locking, firewall, intrusion prevention or payment anti-fraud systems.

Keywords: location-aware RBAC, GeoXACML, mobile security.

1 Introduction

The objective of this paper is to investigate how location-aware RBAC policies can be implemented in the eXtensible Access Control Markup Language (XACML), which is an authorisation policy language [15]. The solution is based on existing profiles for RBAC [3], and the Geospatial eXtensible Access Control Markup Language (GeoXACML) for location based access control [2].

There are numerous application possibilities for location-aware RBAC. It can be useful for automatic locking/unlocking of the phone based on location. The phone can for example be automatically unlocked in the work premises and at home, but not anywhere else. Another possibility is to use location-aware RBAC for mobile payment applications, to handle location-dependent threats for a mobile payment service. Payment may for example not be recommended or permitted in certain areas. This can either be due to threats against the mobile terminal, like the threat of physical theft or the risk of a localised cyber attack against the mobile terminal in a given location, for example at a rogue access point or bluetooth attacks. It can also be because the risk of fraud is considered large in the given location, based on past known incidents. Another strategy is to explicitly allow mobile payments by letting the user authorise a new location by providing his credentials. This can be used to reduce the risk of fraud, since

R. Prasad et al. (Eds.): MOBISEC 2011, LNICST 94, pp. 172–183, 2012.

mobile payments then only would work in locations and with vendors that were explicitly authorised by the owner.

We assume that the exchange of information related to XACML policy administration can be performed securely, for example over an encrypted link with signed messages, to ensure the confidentiality and integrity of the XACML policy management. It is furthermore assumed that the storage and execution environment can be secured using trusted computing or similar techniques. The paper does not go into details on the authentication process, which can be covered using existing methods and protocols, for example based on the Security Assertion Markup Language (SAML).

This paper is organised as follows: The next section describes how the RBAC profile of XACML implements role-based access control. Section 3 introduces GeoXACML, which is an extension of XACML that provides support for fine-grained authorisation based on geographical data types and functions. This section furthermore elaborates on how location-aware RBAC can be implemented using GeoXACML. Section 4 gives an example of a role-based authorisation policy for intrusion prevention systems based on GeoXACML. Section 5 discusses advantages and disadvantages with the proposed solution, Section 6 goes through related work and Section 7 concludes the paper and discusses future work.

2 RBAC in XACML

Subsequent sections assume that the reader has a basic understanding of XACML. In the following, the XACML 3.0 namespace is denoted as *&xacml*;, the XML Schema namespace is denoted as *&xs;* and our own extensions are defined in the namespace *http://www.prile.org:,* denoted by *&prile*; and roles, defined as *&xacml;:subject:role:,* are in short denoted as *&role;*. The GeoXACML namespace *urn:ogc:def:dataType:geoxacml:1.0:* is in short denoted *&geox;*.

The Organization for the Advancement of Structured Information Standards (OASIS) has defined core and hierarchical RBAC profiles of XACML 2.0 [3]. Figure 1 illustrates how the XACML RBAC profile can be implemented in a Service Oriented Architecture. The most noticeable difference, compared to the standard RBAC model, is that the authorisation model is subdivided into three main functions:

- a *Role Enabling Authority (REA)* that is responsible for managing the User Assignment (UA) mapping (users to roles) in the standard RBAC model;
- an *Identity Manager* that both authenticates the users towards different roles, manages user sessions and can send authorisation requests to access given objects or resources towards the XACML Policy Decision Point (PDP);
- the *XACML RBAC profile* that gives authenticated sessions, with a set of enabled roles, access to given objects/resources based on the permissions these roles have in the XACML PDP.

The REA can be combined with the Identity Manager into an Identity Provider or Single Sign-On (SSO) service that provides authentication and authorisation

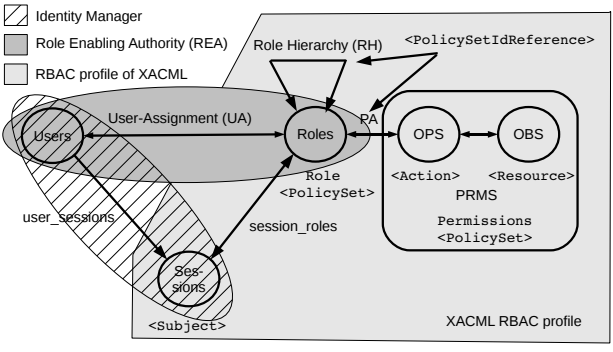

Fig. 1. Implementation of the RBAC model in a Service Oriented Architecture (SOA), based on the RBAC profile of XACML

of users. Examples of such services can be LDAP directory services or federated services like Shibboleth[1] that can use SAML in web services to perform authorisation requests towards the PDP running the XACML RBAC profile. SAML assertions can then be verified by the PEP, presuming that the PEP is trusted. This paper does however not go into the details of the SAML-XACML interaction for space reasons.

The XACML RBAC profile [3] defines how XACML policies can be used to implement the core and hierarchical parts of the NIST RBAC model [9]. This profile expresses:

- *RBAC user_sessions* that are implemented as XACML <*Attributes*> representing authorised users and currently enabled roles for each user in the XACML <*Subject*> elements;
- *RBAC Roles* are expressed as Role <*PolicySet*> elements;
- *RBAC Objects (OBS)* are expressed using XACML <*Resource*> elements;
- *RBAC Operations (OPS)* are expressed using XACML <*Action*> elements;
- *RBAC Permissions (PRMS)* are expressed using XACML Permission <*PolicySet*> elements;
- *RBAC Permissions Assignment (PA)* and *Hierarchical RBAC* are implemented using the <*PolicySetIdReference*> element which refers to other Permission <*PolicySet*> elements.

The XACML Role and Permission policy sets make it easy to extend the permission assignments, to handle location dependent roles or permissions, by adding additional constraints on the respective policy sets. Permissions from different Permission policy sets can be aggregated by including a <*PolicySetIdReference*>

[1] Shibboleth can be found at `http://shibboleth.internet2.edu`

element to each set of permissions that should be included into the policy set of a given role. The role hierarchy is therefore implicitly defined in the XACML RBAC profile by adding <*PolicySetIdReference*> pointers to inherited permissions from other roles, as shown in Fig. 2.

A difference between the XACML RBAC profile and the standard NIST RBAC model, is that the definition of RBAC *Users* and *Sessions* normally are external to the XACML RBAC profile. The RBAC XACML profile presumes that the role(s) and session specified in the XACML request are valid for the given user. Handling of which roles a given user is allowed to enable must instead be done in the REA and sessions are authenticated by the Identity Manager.

An additional restriction in the XACML RBAC profile, is that an RBAC compliant XACML PDP must ensure that Permission <*PolicySet*> instances can never be used as the initial policy of an XACML PDP [3]. This can for example be ensured by keeping Role and Permission policy sets in separate PDP's or by adding a semantic restriction that only Role policy sets can be accessed directly in requests towards the Policy Enforcement Point (PEP).

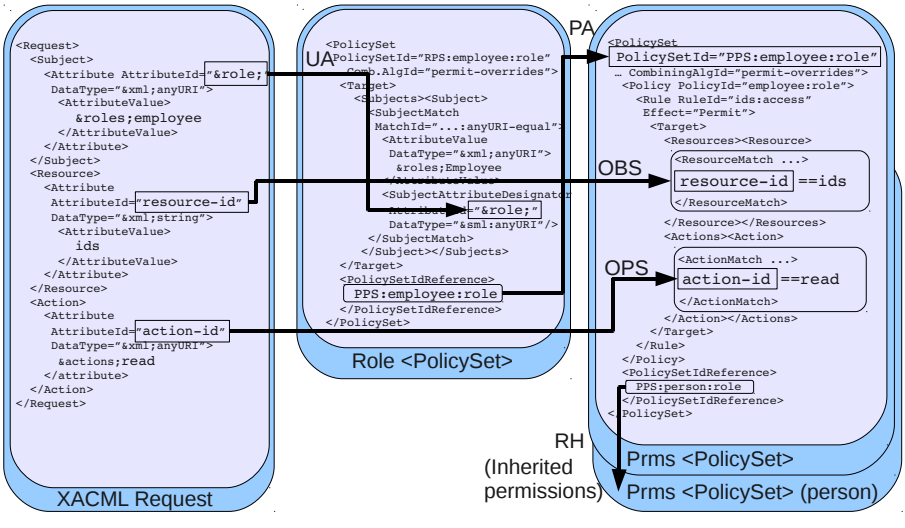

Fig. 2. Inter-dependencies between the XACML Request, the Role policy set and the Permissions (Prms) policy set both for core and hierarchical RBAC

The set of <*Subject*> attributes in the XACML request furthermore contains the *role* or *a set of roles* being enabled by this session. The <*Resource*> attribute of the request refers to the object(s) (OBS) being authorised, and the <*Action*> attribute of the request describes the operation(s) (OPS) that will be performed on the object(s). Fig. 2 shows that the permission assignments (PA) is done by

adding a <*PolicySetIdReference*> pointer to the Permissions <*PolicySet*> that contains the permissions for the given role.

The approach chosen for the XACML RBAC profile will in general provide a lower granularity for PA than the traditional RBAC model, since it refers to a <*PolicySet*> for a given role, instead of providing a true many to many relationship between permissions and roles. The XACML RBAC profile does in other words implement roles, but not strictly according to the NIST RBAC standard. It is perhaps neither viable nor desirable to implement the RBAC standard with finer PA granularity, since the current design allows for some flexibility in the permission handling that does not exist in standard RBAC. It is for example easy to add support for location handling or other constraints on permission level. It may on the other hand not be viable to implement very fine-grained permission handling due to the XML parsing overhead in XACML. This means that the XACML RBAC model is a hybrid between XACML and RBAC.

3 GeoXACML

GeoXACML defines an extension of XACML for spatial data types and spatial authorisation functions. The spatial data types are based on the Geographical Markup Language version 3 (GML3) [14]. These data types and functions can be used to define spatial constraints for XACML based policies [2], which means that it is possible to support declaration and enforcement of access restrictions on geographic information. GeoXACML defines mainly the geometry model for geometric data types in access rules and geometric functions that can operate on these geometric data types.

```
<Condition FunctionId="&geox;geometry-intersects">
  <Apply FunctionId="&geox;geometry-one-and-only">
    <SubjectAttributeDesignator AttributeId="my-position"
      DataType="&geox;geometry"/>
  </Apply>
  <AttributeValue DataType="&geox;geometry">
    <gml:Polygon id="Grimstad" srsName="urn:ogc:def:crs:EPSG:6.6:4326"
     xmlns:gml="http://www.opengis.net/gml">
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList dimension="2">
            58.34 8.59 58.5 8.60 ...
          </gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </AttributeValue>
</Condition>
```

Fig. 3. GeoXACML geographical *Condition* example

3.1 Implementing Location-Aware RBAC Based on GeoXACML

Location-aware RBAC can be implemented by adding constraints in the form of *Conditions* on the role and permission policy sets that evaluate a logical expression based on GeoXACML geographical functions. Fig. 3 shows an example XACML condition that checks whether the device position reported by the *<Subject>* attribute *my-position* intersects with or is contained in the polygon defining the town *Grimstad*. This subject attribute contains the current GPS position of the device, for example represented as a GeoXACML circular polygon with radius equal to the measurement uncertainty. This subject attribute can either be provided by the XACML Policy Information Point (PIP) or it can be passed in as part of the XACML request. The function *geometry-one-and-only* is needed because the *<SubjectAttributeDesignator>* element returns a bag of potentially zero or more elements. This function returns the first element of type geometry, and if there are no such elements or more than one element, then the function returns *INDETERMINATE*.

Adding location dependent constraints on the Permissions *<PolicySet>* gives the possibility to place additional geographical restrictions on inherited permissions. This effectively means that it is possible to add geographical restrictions on any level desired, from a course-grained role level and to a fine grained permissions level. XACML Constraints also make it easy to add permissions that apply everywhere but a given set of locations. For example a firewall rule that is applicable for any position p except an area A can be expressed by inverting the result of the geometrical inclusion test, i.e. $p \notin A$.

It should also be noted that the proposed model also supports temporally aware RBAC policies based on XACML. This can be implemented by adding XACML *<Condition>* constraints based on standard XACML functions for time handling in the Role or Permission policysets.

3.2 Spatial Separation of Duties Constraints in GeoXACML

This section sketches how spatial separation of duties constraints can be implemented in GeoXACML. A deficiency with the OASIS XACML RBAC profile is that it does not support static or dynamic separation of duties (SSD/DSD), as specified in the RBAC standard [3]. The reason for this, is that the REA and Identity Management functions are defined outside the XACML RBAC profile, which means that the assignment of users to roles and management of active sessions are not defined as part of the XACML RBAC profile. This also means that a basic assumption is that the XACML authorisation function must trust the REA in order to implement separation of duties constraints.

SSD constraints mean that a user must not be authorised with conflicting roles [9]. These constraints can be implemented in XACML if the REA exposes the mapping of currently authorised users to roles as XACML attributes via the PIP. This can for example be implemented using a SAML *AttributeQuery* for a given user, which implies that the user name must be passed in as part of the XACML request in order to enforce SSD. XACML *<Condition>* constraints on

the user-role mapping can then be used in the Role <*PolicySet*> to verify static separation of duty constraints by counting the number of enabled roles that are in the set of conflicting roles in the REA for a given user.

DSD puts restrictions on which roles that can be enabled at the same time in the same user session or across different user sessions [9]. DSD can in a similar way be implemented in XACML if the REA exposes the mapping of active user sessions to roles as XACML attributes via the PIP. This means that XACML <*Condition*> constraints on the user session resources from the Identity Manager can be used in the Role <*PolicySet*> to check for conflicting roles within or across user sessions. This will require that a unique session identifier, for example the SAML session ID, is sent as an attribute of the XACML request.

A central lock manager, for example as suggested in [7], will then be required to ensure safe policy checks for concurrent authorisation requests, so that the SSD/DSD constraints ensure that less than a given number of sessions for a given role are active at the same time. This central lock manager needs to be extended to block updates of sessions and their roles by the REA and the Identity Manager during checking of SSD/DSD constraints to avoid race conditions that could violate the constraints.

Spatial SSD constraints (SSSD) furthermore imply that if a user is assigned to a role in one location, the user cannot be assigned to another role in this location if these two roles are conflicting [10]. In a similar way, Spatial DSD constraints (SDSD) ensure that constraints on which roles that can be enabled in the same user session or across user sessions can be enforced for a given location. The above mentioned scheme can easily be extended to support SSSD or SDSD by adding GeoXACML geographical restrictions to the respective SSD or DSD XACML <*Condition*> constraints on the Role <*PolicySet*>.

Implementing dynamic separation of duties constraints for the XACML RBAC profile is not trivial, since it will require both a central lock manager and an extension of the REA in order to expose the mapping of active user sessions to roles. It is also problematic from both security and privacy perspectives that XACML has access to all active sessions for all users, especially if this occurs in a federated environment where the authenticated sessions may belong to other authorisation functions.

4 IPS Policy Example

This section discusses location based permissions for controlling Intrusion Prevention System (IPS) or firewall rules which can be used to provide more flexible protection of mobile terminals. This can be implemented by adding a separate XACML *Action*, for example *configure-ids* or *configure-firewall,* that the IPS or firewall can use to authorise and update their rule sets for a given set of user roles. Some IPS or firewall rules will then be enabled at any position. Other rules may be enabled or disabled in certain locations or for certain roles or combination of roles. This can be done by having a separate threat manager service that

```
<PolicySet PolicySetId="PPS:payment:role"
  PolicyCombiningAlgId="&xacml;policy-combining-algorithm:permit-overrides">
  <Policy PolicyId="IPS:bluetooth:permissions"
    RuleCombiningAlgId="&xacml;rule-combining-algorithm:permit-overrides">
    <Rule RuleId="IPS:payment:permissions:in:Grimstad"
      Effect="Permit">
      <Target>
        <Resources><Resource><AnyResource/></Resource></Resources>
        <Actions><Action>
          <ActionMatch MatchId="&xacml;function:string-equal">
            <AttributeValue DataType="&xs;string">
              configure-ips
            </AttributeValue>
            <ActionAttributeDesignator AttributeId="&xacml;action"
              DataType="&xs;string"/>
          </ActionMatch>
        </Action></Actions>
      </Target>
      <Condition FunctionId="&geox;geometry-intersects">
        <Apply FunctionId="&geox;geometry-one-and-only">
          <SubjectAttributeDesignator AttributeId="my-position"
            DataType="&geox;geometry"/>
        </Apply>
        <AttributeValue DataType="&geox;geometry">
          <gml:Polygon id="Grimstad" srsName="urn:ogc:def:crs:EPSG:6.6:4326"
           xmlns:gml="http://www.opengis.net/gml">
            <gml:exterior><gml:LinearRing>
              <gml:posList dimension="2">
              58.34 8.59 58.5 8.60 ...
              </gml:posList>
            </gml:LinearRing></gml:exterior>
          </gml:Polygon>
        </AttributeValue>
      </Condition>
    </Rule>
    <Obligations>
      <Obligation ObligationId="&prile;ips-rule:apply"
       FulfillOn="Permit">
        <AttributeAssignment AttributeId="&prile;enable:snort:rule:1"
          DataType="&prile;snortrule">
          reject tcp $HOME_NET any -&gt; $HTTP_SERVERS $HTTP_PORTS (\
          msg:"Mobile␣payment␣attack␣rejected␣(fraud␣risk)";
          flow:to_server,established; uricontent:"http://www.mybank.com";
          nocase;  classtype:attempted-recon; sid:10000001; rev:1;)
        </AttributeAssignment>
    </Obligation>
  </Obligations>
  </Policy>
</PolicySet>
```

Fig. 4. Example *Permissions <PolicySet>* for location-aware IDS policy

enables or disables the rules based on location dependent threats. Such a service checks the threat policies and updates IDS and firewall rules continually based on parameters such as time, device speed and threat profile updates.

Fig. 4 shows an example *Permissions <PolicySet>* that is used to configure security requirements for the *payment role* of a mobile device. The *<PolicySet>* requires that the IPS should be installed with a rule that rejects connections to *http://www.mybank.com* if the user is within the Grimstad area, for example to reduce the risk of fraud. This rule will be installed the next time the threat management PEP performs an XACML request on the current user session. If the user session has the *payment role* enabled and the PEP issues a *configure-ips*

action, then the PDP will reply with a *Permit* decision with an *Obligation* to apply the given IPS rule.

If the user later moves outside of the Grimstad area, then this restriction will be removed the next time the threat management PEP is scheduled.

5 Discussion

A challenge with the proposed approach, is that the policy management in the PAP is quite complex. The role hierarchy is for example implicitly defined via links between permissions policy sets in the XACML RBAC profile, something that is not very intuitive. Routine policy generation tasks can however be automated, and the resulting API can be made similar to the functional specification in traditional RBAC [9], however with the necessary adaptations to handle geographical constraints for roles and permissions and also for managing the permissions policyset. The details of the modified API is however beyond the scope of this paper.

A disadvantage with location dependent firewall rules, is that it may not be very user friendly or intuitive. The final user may for example be puzzled if location based policies prohibit access, since it is not clear to the user *why* access is prohibited. This can however be handled by adding an additional XACML *<Obligation>* to the policy that will specify a *reason* for why access is prohibited, for example to give a notice that "Access is prohibited in this location to service XXX due to risk of fraud".

It is also possible to use other means than firewall or IPS rules to restrict access. It would probably be more useful for a mobile payment application if a successful authorisation returns an obligation containing part of a cryptographic key needed to gain access to the service, rather than managing firewall or IPS rules. In particular on phones where the firewall or IPS run as untrusted applications, since these then easily can be disabled by a determined adversary that has stolen or hacked into and compromised the phone.

An issue that needs to be considered is the reliability of the positioning system, and whether positions easily can be forged[2] in order to avoid location dependent access restrictions. GPS is for instance relatively accurate in an outdoor environment, but it quickly loses connection indoors.

Another problem is that GPS data coming from the device itself is not guaranteed to be correct if the operating system is compromised. For example if a rootkit or rogue GPS device driver intercepts and changes the GPS signals on the fly, in order to influence the authorisation process.

This risk can be reduced to some extent using traditional mitigations like AntiVirus and good patching and update procedures. It is however more efficient to use techniques like cryptographically signed drivers and secure booting of the phone operating system where a Trusted Platform Module (TPM) in future mobile phones ensures that unauthorised modifications, for example by a rootkit,

[2] GPS Spoofing Countermeasures `http://www.homelandsecurity.org/bulletin/Dual%20Benefit/warner_gps_spoofing.html`

causes the TPM to refuse booting the phone [8]. The risk of being infected by a rootkit can probably not be eliminated completely, since there always will be a risk of stack or buffer overflow vulnerabilities or other flaws also in trusted and signed applications.

This means that using a positioning technique like GPS has its deficiencies when used for access control purposes. These deficiencies will also apply if the access control is used to manage location dependent threats. It is in some cases hard to put a clear distinction between areas that are safe and areas that are unsafe and if a safety margin is added to location threats, then this may harm legal businesses which clearly is undesirable. Using near-field communications (e.g. RFID) together with a *proof-of-location* protocol is a promising technology that can be used as an additions means to verify known locations [12]. This can improve the precision of location-aware RBAC and also reduce the risk of harming legal businesses.

6 Related Work

This paper describes how location-aware role based access control can be implemented by combining GeoXACML and the RBAC profile of XACML. Such a GeoXACML-based RBAC solution has to the best of our knowledge not been published before.

There are numerous previous works on location-based or spatial RBAC models. This discussion does not cover all RBAC models, but compares our solution to some of the well known traditional location based RBAC solutions. SRBAC is one of the earlier models of a spatial RBAC system [10, 11], and it has been suggested to define an XACML-based location aware RBAC model based on SRBAC [1]. Our model has moved the location constraints into the Role and Permissions (PRMS) definitions instead of expressing location references as an explicit relationship between *Roles-Locations* and *Locations-Operations* as SRBAC does.

GEO-RBAC extends the traditional RBAC model with spatial entities that are used to model objects, user positions and geographically bounded roles [4, 6]. A model for enforcing spatial constraints for mobile RBAC systems based on GEO-RBAC is described in [12]. Roles in GEO-RBAC are enabled based on the position of the user. GEO-RBAC does however not support defining location on object permissions. LRBAC and LoT-RBAC solve this deficiency [13, 5].

The nice feature with all native RBAC models, is that they are fast, efficient and have well defined formal definitions founded in set theory. Extensibility for these models has however evolved over time, and is not yet standardised for location or time based RBAC models. XACML is on the other hand designed from the start to be extensible, with rich semantics for defining constraints. The RBAC profile of XACML implements role based access control, however not with the same level of granularity as native RBAC solutions and also with deficiencies when it comes to enforcing separation of duties constraints. The extensibility of XACML makes it easy to embed and combine other XML-based

standards, like in this case the RBAC profile of XACML and GeoXACML, and it also makes it trivial to add arbitrary constraints both on roles and permissions. An advantage with our model is that the physical location definition is embedded in the XACML policies, which simplifies deployment of location-based policies. This is useful for outsourced managed security services, where updated threat profiles then can be generated and deployed according to the needs of the mobile terminal.

7 Conclusions

This paper demonstrates how location-aware RBAC can be implemented in GeoXACML, based on the XACML RBAC profile. It furthermore shows how static and dynamic separation of duties constraints can be implemented for this solution. An advantage is that it allows for embedding geographical information directly into the authorisation policies, instead of using logical addressing of locations.

A potential disadvantage with an XACML based RBAC solution, is that it may scale more poorly than a traditional RBAC solution, especially if a large number of roles and permissions are involved. The main problem in XACML will then be the parsing overhead of XML documents. This can to some extent be mitigated using a decision cache under the presumption that location threats do not change too rapidly.

The general discussion also shows that it can be difficult to put a clear distinction between areas that are safe and areas that are unsafe due to the inherent unreliability of positioning systems and also unreliability in the definition of what areas are considered unsafe. This may harm legal businesses which clearly is undesirable. There are in other words several concerns with location-based threat management and location-based authorisation.

Future work includes implementing and testing the location-aware RBAC solution, including to elaborate and demonstrate how well the proposed solution works for static/dynamic spatial separation of duties. There is also more work required on how to reliably identify locations and properly secure the overall solution.

Acknowledgments. This work is funded in part by Telenor Research & Innovation under the contract DR-2009-1.

References

1. Aburahma, M., Stumptner, R.: Modeling location attributes using XACML-RBAC model. In: MoMM 2009, Kuala Lumpur, Malaysia, p. 251 (2009)
2. Matheus, A. (ed.): OGC 07-026r2 Geospatial eXtensible Access Control Markup Language (GeoXACML) version 1.0 (2007), `http://portal.opengeospatial.org/files/?artifact_id=25218`

3. Anderson, A. (ed.): Core and hierarchical role based access control (RBAC) profile of XACML v2.0 (2005), http://docs.oasis-open.org/xacml/cd-xacml-rbac-profile-01.pdf
4. Bertino, E., Catania, B., Damiani, M.L., Perlasca, P.: GEO-RBAC: a spatially aware RBAC. In: ACM SACMAT, p. 37 (2005)
5. Chandran, S.M., Joshi, J.B.D.: *LoT-RBAC*: A Location and Time-Based RBAC Model. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 361–375. Springer, Heidelberg (2005)
6. Damiani, M.L., Bertino, E., Catania, B., Perlasca, P.: Geo-rbac: A spatially aware rbac. ACM TISSEC 10(1), 2 (2007)
7. Dhankhar, V., Kaushik, S., Wijesekera, D.: XACML Policies for Exclusive Resource Usage. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 275–290. Springer, Heidelberg (2007)
8. Dietrich, K., Winter, J.: Implementation Aspects of Mobile and Embedded Trusted Computing. In: Chen, L., Mitchell, C.J., Martin, A. (eds.) Trust 2009. LNCS, vol. 5471, pp. 29–44. Springer, Heidelberg (2009)
9. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuch, D.R., Chandraouli, R.: Proposed NIST Standard for Role-Based Access Control (2001)
10. Hansen, F., Oleshchuk, V.: Spatial role-based access control model for wireless networks. In: IEEE 58th VTC, pp. 2093–2097 (2003)
11. Hansen, F., Oleshchuk, V.: SRBAC: A spatial role-based access control model for mobile systems. In: NORDSEC, pp. 129–141 (2003)
12. Kirkpatrick, M.S., Bertino, E.: Enforcing spatial constraints for mobile RBAC systems. In: ACM SACMAT, pp. 99–108. ACM (2010)
13. Ray, I., Kumar, M., Yu, L.: LRBAC: A location-aware role-based access control model. In: Information Systems Security, pp. 147–161 (2006)
14. Cox, S., et al. (eds.): OGC 02-023r4 OpenGIS Geography Markup Language (GML) Encoding Specification Version 3.00 (2002), https://portal.opengeospatial.org/files/?artifact_id=7174
15. Moses, T. (ed.): OASIS eXtensible Access Control Markup Language (XACML) Version 2.0 (2005), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf